APPLICATION NOTE

AP-253

September 1985

Adding Value to Intel's NDS-II Development System Network with CP/M-80

BRIAN VALENTINE
DSO APPLICATIONS ENGINEERING

Order Number: 231533-001

INTRODUCTION

Word processing has long been a desired, if not necessary, feature of a microprocessor development system. Although most systems can support high-level languages, in-circuit emulators, and many other tools to develop microprocessor software and hardware, few support advanced word processing.

A typical microprocessor design project consists of three general phases: design and staffing, implementation, and test and integration. Word processing plays a major part in all three phases. Each phase requires an advanced editor, such as Wordstar by Micropro, to generate the supporting documentation that accompanies a project.

Microcomputer Development Systems for the design engineers should have the capability to access word processing tools. This feature would eliminate the need for a second system or terminal on the engineer's desk. The engineer also should have the ability to share his or her word processing database with other engineers involved in the project.

The design and staffing phase also requires electronic spreadsheets, such as Multiplan by Microsoft, to track potential problems or determine the number of engineers required for the project. The typical engineering support staff are all running upon various different systems. This application note will show how added capabilities to Intel's Network Development System (NDS-II) will provide each engineers workstation with word-processing and spreadsheet capabilities. In addition, since all workstations are connected via the NDS-II and Ethernet, each engineer will have access to other engineers' word-processing databases.

The NDS-II Network

Intel's NDS-II enables development system mainframes to be connected into a network using Ethernet. Additionally, each mainframe can host several ISIS clusters that use low-cost serial lines to support terminals. The complete product line is described in the NDS-II System Description (See Appendix C for complete details).

Low-cost terminals allow everyone to share and manipulate files and data directly on the network. This feature eliminates many intermediate steps required in producing a final document. The addition of CP/M-80 for the NDS-II, combined with word processing and spreadsheet programs, further increases word-processing efficiency. Figure 1 shows a typical NDS-II environment.

SOLUTION

The problem is to add word-processing capability to the NDS-II network. Since CP/M-80 already runs on an Intel Series-II development station in standalone mode, the solution is to modify CP/M-80 to access remote files located on the NDS-II file server instead of local floppies. This solution will provide networking to CP/M-80 and also increase the I/O performance, since all files are now accessed over Ethernet, rather than from slower floppy disk drives.

The method chosen to modify CP/M-80 is to modify the BIOS of CP/M-80. (For more information, see "CPM Alteration Guide," (c) 1979 Digital Research and Appendix C.)

OVERVIEW OF CP/M-80

CP/M-80 contains five sections:

- BIOS – basic I/O system
- BDOS – basic disk operating system
- CCP – console command processor
- TPA – transient program area
- SPA – system parameter area

Figure 2 shows where each of the sections reside in memory on a Intel Series II station. Each section executes specific functions. The BIOS section is responsible for the I/O options to all the system-specific peripherals. When CP/M-80 is ported from one system to another, the BIOS is the only section that changes (length: variable).

The BDOS section is responsible for the general non system-specific operation of the peripherals. BDOS functions call BIOS functions to operate the peripherals. The BIOS and BDOS are collectively know as the full disk operating system (FDOS) (length: fixed).

The CCP section reads commands from the user. It has the following built-in commands: DIR, ERA, REN, SAVE, TYPE and USER (length: fixed).

The TPA section is used as user program space. Application programs are loaded and operate in this section (length: available memory – size of BIOS, BDOS and SPA). Note: If needed, the TPA can occupy space up to the beginning of the BDOS. If this happens, CCP must be reloaded when the program running in the TPA exits.

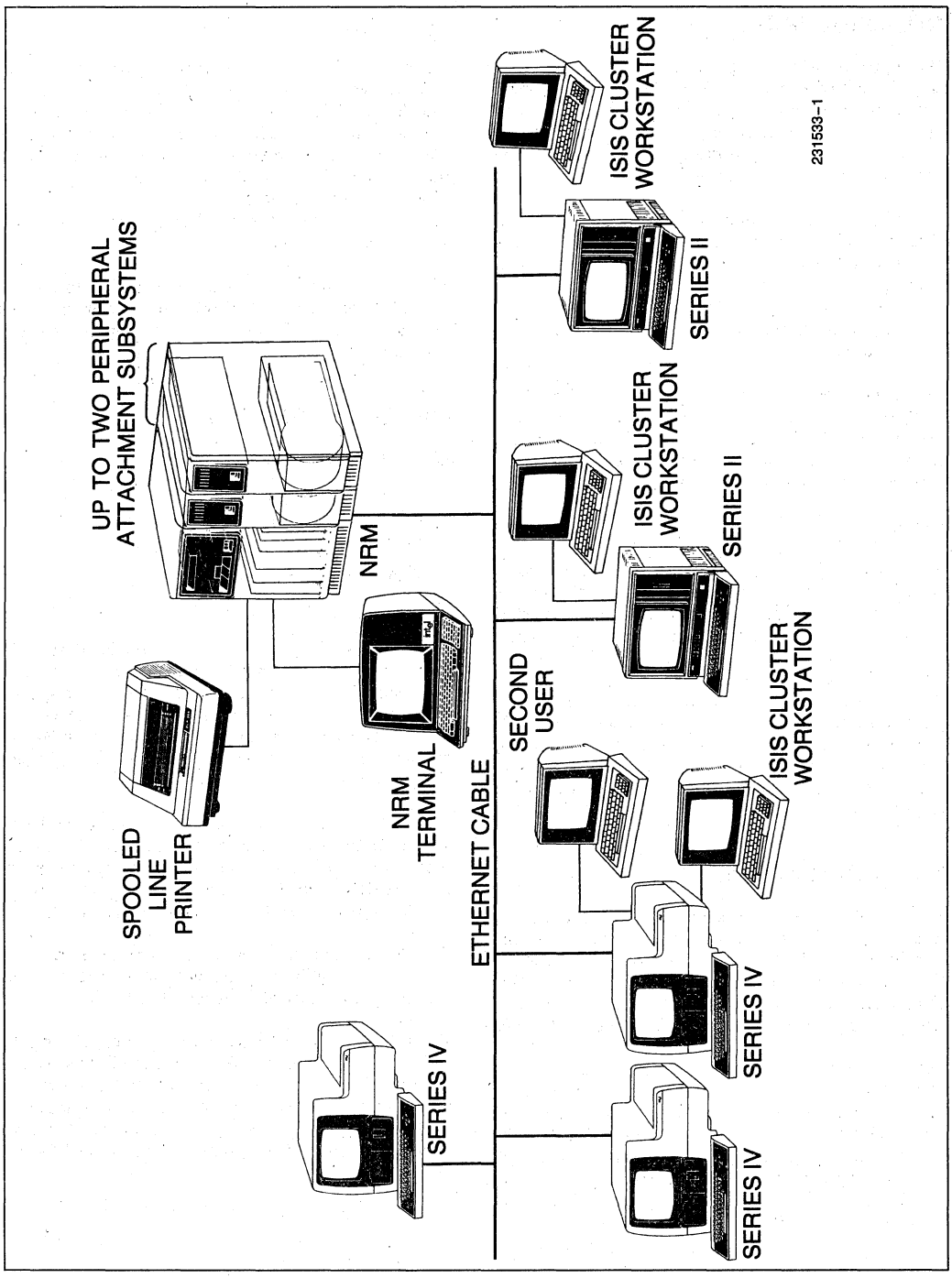

Figure 1. NDS-II Configuration

The SPA section contains operating system information such as the current disk, user number, peripheral assignments, the start address of the BIOS and BDOS sections, restart locations and the default buffer. Figure 3 shows a breakdown of the SPA (length: 256 bytes).

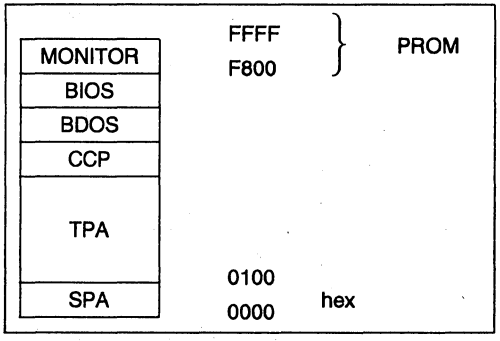

Figure 2. Series II Memory Map of CP/M-80

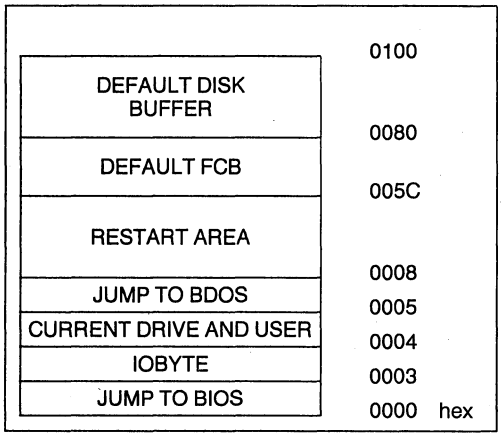

Figure 3. Breakdown of SPA

BIOS FUNCTIONS

Now that CP/M-80 has been broken down, the problem is how to port the Series II standalone version of the BIOS to process all I/O requests over the NDS-II network instead of local floppies. Due to legal reasons, BDOS and CCP cannot not be changed; therefore, we must modify the BIOS only.

The BIOS can be broken down into three sections:

- A jump table to BIOS functions
- Disk parameter/description blocks
- The code to execute the BIOS functions.

The jump table contains all the entry points into the BIOS functions. The disk parameter section contains blocks that describe how each disk (A:–P:) is formatted, for example, disks can be formatted as single density, double density, or Winchester. The last and largest section of the BIOS is the code that executes the BIOS functions.

In Figure 3, location 0000 hex in the SPA stores the address of the first position (or function) in the BIOS jump table. Therefore, any program running under CP/M may access all the BIOS functions by using the jump vector stored at this address.

NOTE:
Any program accessing the BIOS jump vector must access it with a CALL statement, not a JMP statement, since all BIOS functions will use a RET statement to return to the calling program.

The following list of functions are contained in the BIOS and accessed via the BIOS jump table:

- Initial cold start
- Warm boot
- Console status
- Console input
- Console output
- Printer output
- Punch output
- Reader input

Beginning of disk routines

- Home – Restore current disk to A:
- Select disk – select a particular disk to be the current disk
- Select track – select a track in the current disk
- Select sector – select a sector in the current disk/track
- Select disk buffer address – address of where to read/write the sector from/to the disk to/from memory
- Read sector – read the current sector from the current track on the current disk into memory starting at address stored via the BIOS function number 13
- Write sector – write the sector from memory to disk

- List status
- Convert current logical track/sector into real track/sector

NOTE:
If you are modifying the BIOS, all functions must be supplied.

The Series II standalone version of BIOS currently uses the above functions to access local floppy drives. To modify the BIOS to access NDS-II files, you do not need to change the first eight functions. However, the remaining functions process I/O to the local floppies and must be changed to access I/O over the NDS-II network.

NETWORK CP/M DISK IMAGES

The first problem that must be solved is the accessing of disks. Since Network CP/M does not use the local floppies, a similar storage device(s) must be supplied at the NDS-II file server (NRM).

The NRM Winchester drives are formatted using an extended iRMX™86 structure. This operating system is not similar to CP/M-80. Each NRM drive has a hierarchical file structure and supports multiusers, with individual home directories. Disk storage at the NRM may total four 84 Mb drives. The problem is how to set up a file or disk system on the NRM that will allow CP/M-80 file/disk structures and also support multiuser access to these CP/M-80 disks.

The first design decision in porting CP/M-80 to the NDS-II was to limit Network CP/M-80 to four disks (A:–D:). This decision is two-fold:

- Standalone CP/M-80 only supports A:–D:
- Network CP/M BIOS only has space for four disk parameter blocks.

It was also decided to store these disks at the NRM as data files. These data files will be structured as if they were CP/M floppy disks. Therefore, when CP/M-80 thinks it is accessing a floppy, by track and sector, it will really be accessing a file on the NRM. When a CP/M program wants to read a sector on the disk, our modified BIOS will instead seek into the NRM file to the position where the desired sector is stored and read the sector from the file.

This design creates more problems that must be addressed. The first is the format of the "disk images". The second is the location on these disk images on the NRM to give multiuser support for Network CP/M.

The disk images will be similar to a double density CP/M floppy. Figure 4 shows a layout of a Network CP/M disk image. Network CP/M disks have the following design:

- Each disk has 254 tracks
- Tracks 0 to 3 are reserved for the CP/M operating system
- Track 4 is reserved for directory
- Track size is 2K
- Block size is 2K
- Sector size is 128 bytes
- There are 16 sectors per track
- There is a Maximum of 64 directory entries
- Total disk size is ½ Mbyte

Since the BIOS contains the disk parameters blocks, we must supply four parameter blocks (A:– D:) in the Network CP/M BIOS. These blocks are all identical and describe each disk with the above format.

NOTE:
The Network CP/M utility MAKDSK is used to create disk images. The disk image is created with only the first five tracks (system and directory). As data is added to the disk image, the disk image file will grow by the size of the data. This space-saving feature allows for many CP/M disk images to reside on the NRM file structure, taking only as much disk space as the valid data they contain.

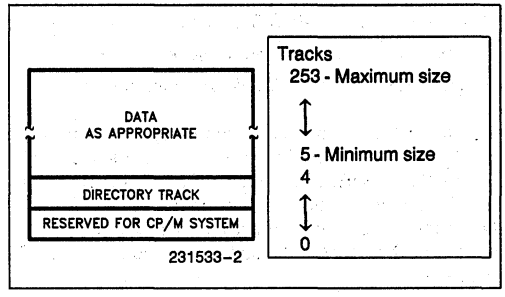

Figure 4. Disk Image Format

Multiuser access to data can be sub-divided into two problems. The first concerns storing all the system files, so that only one copy of the files is on the system, and all users have access to this copy. System files are files/programs like STAT, PIP, Wordstar, etc. The second problem is that users must have access to private data.

A specific design decision solved these problems. A: disk image will be read only and contain all system files. Therefore, each user on the network will access the same A: disk. B:–D: disks will be private and unique to each user, will store private data, and will have read/write access.

At the NRM all ISIS users access a system directory called ISIS.SYS. This directory is assigned to device :F0: on each workstation and contains all the ISIS system files. Therefore, this is a logical place to store the CP/M disk image A:. The name adopted for the disk image is :F0:ADISK.CPM.

The NRM assigns a directory, called the user's home directory, to each user. This home directory, unique to each user, is assigned to device :F9: on the workstation when the user logs on to the network. Therefore, the home directory is a logical choice to place the disk images B:–D:. The names adopted for B:–D: are:

B:–:F9:BDISK.CPM
C:–:F9:CDISK.CPM
D:–:F9:DDISK.CPM

We have now solved the Network CP/M disk image problem by designing the disk image format and allowing multiusers to access these disk images.

LOADING CP/M OVER ISIS

Normally when a Series II or ISIS cluster is powered on—and boots from—the NRM, the workstation is booted with ISIS. Therefore, our next problem is when running ISIS on the workstation to load CP/M into the workstation memory from disk image A:. We also must be able to restore ISIS when the user is finished with CP/M and wiches to return to ISIS mode.

CP/M-80 is similar to ISIS if both are displayed in their memory map diagrams. Although both operating systems contain the same sections, (see Figure 2 and Figure 5) ISIS and CP/M-80 store the sections in different parts of memory.

In Figure 5, the CLI (command line interpreter) is equivalent to the CP/M CCP region. The ISIS CLI, however, is loaded into the TPA. A command loaded by the CLI starts at address 3680 hex. Therefore, the ISIS CLI is overwritten with the command. When the command or program exits, ISIS will reload and CLI into the TPA region.

Also, a region called the OVERLAY REGION begins at address E800 hex. This area of memory is used by a program that supports overlays. A program that does not require overlays may use memory up to the SPA.

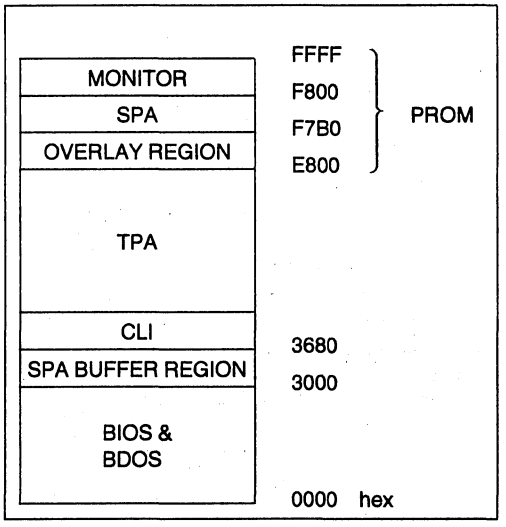

Figure 5. ISIS Memory Map

The problem of how to load CP/M from the disk image A:, saving ISIS, and reloading ISIS when the user is ready to return to ISIS now can be solved. First, create an ISIS program called LOADCPM. This program is a CP/M-80 loader, which:

- Opens file connections to the four CP/M disk images. A: is opened read only; B:–D: are opened read/write access. Note:These disk images are stored at the NRM as normal data files.
- Opens a fifth file called :FO:ISIS.SAV.
- Saves the contents of memory from address 0000 hex to 8FFF hex, and from F000 to FFFF. This is done by writing the contents of the memory into the file ISIS.SAV. This saves the current status of ISIS and the LOADCPM program that is running in the TPA. The ISIS program LOADCPM will not consume the TPA past the address 8FFF; therefore, to save the program the address 8FFF was selected as the upper bound.
- Loads CP/M now that ISIS is saved. The next step is to load the CP/M BIOS, BDOS, and CCP into high memory from the disk image A:.
- Stores the file connections for the disks A:–D: into a BIOS table. This allows the BIOS to read or write to these connections. Therefore, when a CP/M program requests data from disk, the network BIOS will use the file connections to seek into the disk image and read/write the sector.
- Turns control over to CP/M. This is done by calling the first address in the BIOS jump table (cold boot vector). BIOS will first store the return address to the ISIS program called LOADCPM and then turn control over to the CP/M CCP. At this point, the CP/M prompt A will be printed on the console. The user is now running CP/M.

NOTE:
When the last step is finished, the user's workstation (Series II or cluster) is running the CP/M operating system. All CP/M commands/programs are available. Network CP/M is not a CP/M simulator that runs on top of ISIS.

The second part of the solution is how to restore ISIS when the user is ready to return to ISIS. This is done by creating a CP/M program called ISIS.COM. The program operates as follows:

- The program ISIS.COM will reload ISIS and the ISIS LOADCPM program from the ISIS.SAV file.
- At this point, the contents of memory are exactly equal to the point at which the LOADCPM program saved it. ISIS and the program LOADCPM are restored. Now, ISIS.COM will load the PC with the return address to the LOADCPM program, which was saved by the BIOS at the start-up of CP/M. This effectively does a return to the LOADCPM program.
- LOADCPM closes all the disk image files and exits. The exit loads the ISIS CLI, and the user is running ISIS.

Finally, to make the system easier to use, the LOADCPM program is renamed to CP/M. The ISIS user who wishes to run Network CP/M enters the ISIS command: -CPM and when finished, enters the CP/M command: A>ISIS

NETWORK CP/M UTILITIES

Specific ISIS and Network CP/M utilities are supplied with the Network CP/M package. They were developed either as a tool in writing Network CP/M or as a necessary utility needed by the Network CP/M user.

ISIS programs are MAKDSK, ADDSYS, CDIR, CCOPY, CP/M, SUCPM, CPMOMF

Network CP/M programs are ISIS.COM, SPOOL.COM

MAKDSK allows the user to create Network CP/M disk images. The image created will be an empty/non-system CP/M disk.

Command syntax:
```
MAKDSK <CPM_DISK>
```

Example:
```
MAKDSK C:
```

ADDSYS allows the user to add the CP/M operating system to the disk image A: created using MAKDSK. This makes the disk image a CP/M system disk. ADDSYS requires that CPM60.COM and BIOS be located in the ISIS.SYS directory on the NDS-II.

Command syntax:
```
ADDSYS
```

Example:
```
ADDSYS
```

CDIR allows an ISIS user to list a directory of a Network CP/M disk image or a CP/M-80 diskette in Drive 1 of a Series II with a 720 drive.

Command syntax:
```
CDIR <CPM_DISK>
```

Example:
```
CDIR E:
```

CCOPY is a utility that allows an ISIS user to read or write to/from CP/M disk images from/to ISIS files. CCOPY can be used to read data files from the Network CP/M disk images to ISIS files, so that they may be copied to the spooler, etc.

Command syntax:
```
CCOPY READ <CPM_FILE> TO <ISIS_FILE>
     WRITE <ISIS_FILE> TO <CPM_FILE>
```

Examples:
```
CCOPY READ    B:MY.DAT     TO :F1:MY.DAT
CCOPY READ    E:STAT.COM   TO :F1:STAT.COM
CCOPY READ    A:PIP.COM    TO PIP.COM
CCOPY WRITE :F1:STAT.COM TO A:
CCOPY WRITE PIP.COM        TO A:PIP1.COM
CCOPY WRITE :F1:MY.DAT     TO B:MYSTUFF.DAT
```

CPM is an ISIS program that loads Network CP/M onto the Series II or ISIS cluster.

Command syntax:

```
CPM [<C_DISK> [ <D_DISKGT ] ]
```

Examples:

```
CPM
CPM :F4:BDISK.CPM
CPM :F4:BDISK.CPM :F5:BDISK.CPM
```

SUCPM is identical to the CP/M utility, except that A: disk is opened for read/write access. This program should have SUPERUSER access rights only. Only one user may run SUCPM. While the user is running CP/M via SUCPM no other users will have access to Network CP/M. The SUPERUSER can use this program to delete files from, or add files to, A: while running Network CP/M.

CPMOMF allows the ISIS user to convert a program developed under ISIS (ie. compiled by PL/M-80 or assembled by ASM-80) to a CP/M executable program. The program compiled/assembled will be written knowing the CP/M environment.

Command syntax:

```
CPMOMF source_file TO destination_file
```

Example:

```
CPMOMF SPOOL TO SPOOL.COM
```

SPOOL.COM is a CP/M program that copies a CP/M file to the NDS-II spooler. This utility enables the CP/M user to obtain listings quickly and efficiently. Note: The CPM LST: is not supported in Network CP/M. Listings should be directed to a disk file then spooled to the Network printer.

MODIFYING THE BIOS

At this point, you should have a good idea of how the Network CP/M BIOS accesses NRM disk images instead of local workstation floppy drives. The strategy used to modify the BIOS is the following:

- Use the existing workstation standalone version of the BIOS
- Remove the code used in the BIOS to access the local floppies
- Add the code in the new BIOS to access disk images instead of local floppies.

Due to Intel Corporation Proprietary Information used and contained in the Network CP/M BIOS, the code for the BIOS cannot be released.